Supplementary Material for POINTWORLD: Scaling 3D World Models for In-The-Wild Robotic Manipulation

Wenlong Huang[1,†], **Yu-Wei Chao**[2], **Arsalan Mousavian**[2], **Ming-Yu Liu**[2]
Dieter Fox[2], **Kaichun Mo**[2,*], **Li Fei-Fei**[1,*]

[1]Stanford University [2]NVIDIA
[*] Equal advising [†] Work done partly at NVIDIA

A. Appendix

Appendix Contents

A.1. Unseen Rollouts from a Single Pre-trained POINTWORLD

Figure 1. **Unseen rollouts from a single pre-trained POINTWORLD across diverse domains**, visualized with Viser [1]. Given RGB-D captures, POINTWORLD predicts 10-step point flows conditioned on robot flows. We show first prediction, last prediction, and last ground-truth. Green points in GT mark regions occluded during 2D point tracking, for which we observe model predictions are often more accurate because these points are not being supervised in model training. Due to grid downsampling (1.5cm) applied to all point clouds, all 3D visualizations are upsampled to image resolution for visual clarity via nearest neighbors. Interactive visualization is provided in the supplementary material.

A.2. Chunked Prediction

We ablate 10-step chunked prediction against two autoregressive baselines: (i) teacher-forcing (GT input each step) and (ii) self-feeding with 10k warmup steps, plus sliding-window inference ($W{=}1, 5$) using the same chunked model. Teacher-forcing outperforms self-feeding when training and inference strategies are aligned. Evaluating a chunk-trained model with $W{=}1$ (equivalent to self-feeding) incurs the strongest performance degradation; $W{=}5$ recovers some accuracy but degrades after the trained window. Matching chunked prediction in training and testing over the full horizon minimizes drift while amortizing compute with a single forward pass (vs. 2–10 autoregressive), making chunking both more accurate and more efficient.

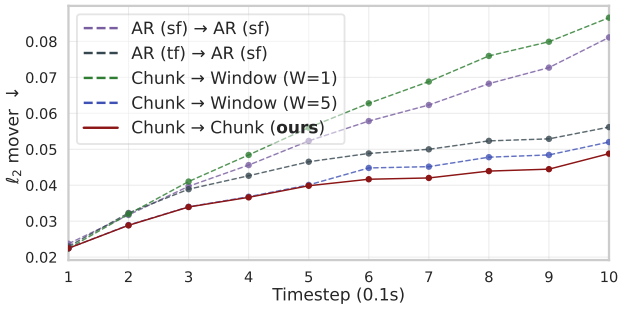

Figure 2. **Ablation on Chunked Prediction**, where we study different rollout strategies in training and testing. Chunked rollouts at both training and inference time lead to significantly less drift than other baselines while amortizing compute with only a single forward pass of the model.

A.3. Partial Observability

We train four variants that differ only in camera count for RGB-D observations: one, two, three, or a random draw of up to three cameras. We then evaluate all models on three settings with up to three cameras. Error on moving points stays sub-centimeter with negligible standard errors, but using more cameras at train time consistently reduces error at test time. Models trained with fixed camera count perform better when more cameras are available at inference. The random-view model is most robust across all test camera counts, suggesting that exposure to varied observability helps the model infer objectness and physical properties under partial observability at inference time.

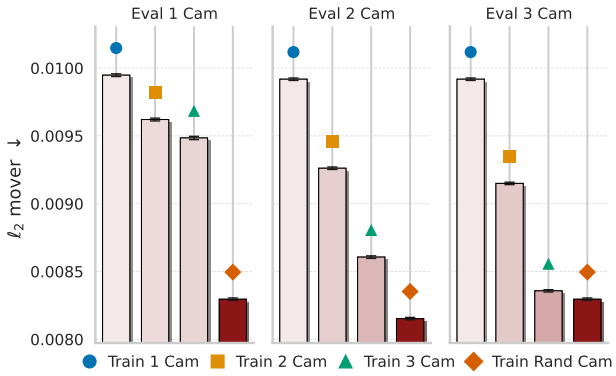

Figure 3. **Ablation on Partial Observability**, where we train variants of PointWorld with varying number of cameras and evaluate them on all settings at test time. PointWorld is robust to different levels of partial observability and benefits from additional cameras in both training and inference. Training with randomized camera counts yields the best performance across all test settings.

A.4. Extended Discussions on Limitations

Static Initial State. The model takes as input RGB-D point cloud together with a finite-horizon sequence of robot actions and predicts how points will move in response. Because no prior frames or velocities are provided, this formulation assumes that the world is static at the observation instant. Supporting fully dynamic initial conditions would require augmenting the input with externally tracked trajectories or recurrent state, which we leave as future work.

Reward/Cost Specification for Action Inference. In this work, we explore POINTWORLD's use case for action inference in manipulation by integrating it with a sampling-based planner, MPPI [2], which requires an explicit specification of reward/cost functions in the same state-action space of 3D point flows. For the scope of this work, we restrict ourselves to manual specification (e.g., moving a subset of points to target locations). Future work may consider automatically specifying (single or multi-stage) reward via vision-language models [3], or inferring reward from demonstrations using inverse reinforcement learning [4], while keeping POINTWORLD as the dynamics function. In addition to planning, action inference can also be alternatively performed by learning a parameterized policy by interacting with the model as the environment via reinforcement learning [5].

Fine-Scale Objects and Calibration Noise. Thin or very small objects (e.g., pens or cables) are challenging to annotate accurately in 3D: modest depth or extrinsic errors can be comparable to the object thickness and lead to ambiguous separation between robot and scene points during ground-truth annotation. In such scenarios, mis-registrations in the ground-truth flows propagate into training and can cause the model to confuse overlapping motions between the gripper and nearby scene points. Improved calibration and depth estimation could strengthen supervision for these fine-grained interactions.

Correlation vs. Causation. Given an observed context frame and a sequence of robot actions, POINTWORLD is trained to predict the subsequent sequence of scene states. As such, it primarily captures correlations present in the training distribution between robot actions, robot motion, and observed scene evolution. In settings where exogenous factors (such as other agents or environment changes not controlled by the robot) also influence the future, these influences are entangled with the robot-induced effects in the data and are not disentangled as separate causal mechanisms. Our experiments therefore evaluate predictive fidelity and planning performance under the observed action-conditioned distribution, rather than claiming to recover the underlying causal structure of the environment.

Lack of Photometric Dynamics. POINTWORLD only outputs displacements of 3D points captured from RGB-D inputs, which focus on geometry and physical interactions rather than appearance. While often visually plausible when rendered as point clouds, it is insufficient if one desires to reason about photometric changes of the environment caused by the robot actions, such as lights or screens turning on and off. Combining POINTWORLD with appearance models that predict emitted radiance, such as those from Gaussian Splatting [6] or Neural Radiance Fields [7], may be necessary for tasks where such photometric dynamics are critical.

Rigid-Body Robot Assumption. Robot embodiment is represented as a kinematic tree of rigid links, and we propagate a fixed set of robot surface points by forward kinematics. This ignores deformations of soft, tendon-driven, or compliant structures (e.g., fin-ray grippers) and non-rigid effects of the robot body. As a result, POINTWORLD reasons only about how the scene moves in response to a forecasted robot geometry, rather than how contact may reshape the robot itself. Extending the representation to include deformable links [8] would enable reasoning about how contact deforms the robot body and how those deformations, in turn, affect contact geometry.

Actuation and Tracking Assumptions. Our formulation treats the robot trajectory as a known, *fully realized* sequence of joint configurations and uses forward kinematics to construct robot flows. As a result, POINTWORLD effectively models "what the environment does if the robot body follows this path," rather than "whether and how the robot will actually realize this path" under a particular controller, actuation limits, or contact-induced tracking errors. This quasi-static, kinematic view of the robot action representation can break down for underactuated or compliant joints (e.g., tendon-driven or compliant hand fingers), or when strong contacts, payloads, or controller changes induce large tracking errors. Extending the method to jointly reason about robot and scene dynamics is an important avenue for future work.

Lack of Explicit Physics Priors. The current formulation is purely data-driven and does not incorporate explicit physics priors such as Newtonian mechanics or conservation-law constraints, to provide a focused study on scaling 3D world models without priors of objectness and of their material and physical properties. Despite this, we observe that POINTWORLD captures many aspects of rigid, articulated, and deformable behavior from data alone. Incorporating physics-informed regularization or hybrid simulators [9] could further improve generalization and extrapolation, particularly in regimes that in-domain interaction data can be collected for accurate scene/object reconstruction, not only for their geometries but also for their physics parameters.

A.5. DROID 3D Annotation Pipeline

DROID [10] is a large-scale robot manipulation dataset with human-teleoperated interactions collected with a wrist-mounted camera and two externally-mounted cameras (randomized over the left and right sides of the workspace). We use all DROID episodes for which raw camera streams are available, irrespective of task success or failure, since 3D world modeling depends only on the observed interactions rather than the task-specific outcomes of the manipulation. Each episode provides stereo RGB streams with ground-truth camera intrinsics for all three cameras, plus robot joint states and a known kinematic model of the robot. In this work, we use the recovered 3D scene flows from the two externally-mounted cameras. All data share a synchronized timestamp. We augment the robot model to include the Robotiq 2F-85 gripper and the custom camera mount used in the standardized DROID setup. For each scene, the pipeline aligns timestamps to a reference stream (binary search to nearest), downsamples by 2, then runs: (i) dense metric depth; (ii) external-camera extrinsic refinement by aligning rendered robot mesh to observed depth; (iii) 2D tracking under workspace and robot masks; (iv) 3D trajectory reconstruction, slicing, and postprocessing. Note that we do not store robot point flows because those can be reconstructed efficiently at training/inference time given known robot URDF and the given joint actions.

A.5.1. Depth Estimation

Per-view metric depth is obtained using a high-quality stereo estimator, FoundationStereo [11]. Note that unlike typical sensor depth, the estimated depth from FoundationStereo does not have a minimum depth threshold for valid depth perception. However, it is still observed that its estimated depth can be inaccurate for distant, especially texture-less, regions (such as walls). Therefore, depth values are sanitized by clamping to a trusted range $[0, 4]\,\mathrm{m}$ and producing a per-pixel validity mask, which is also propagated to 3D points as a per-point depth-valid flag.

A.5.2. Camera Pose Estimation

We do not use dataset-provided extrinsics. Instead, we compute camera extrinsics using a two-stage procedure that leverages the accurate metric depth obtained from FoundationStereo discussed previously. First, we initialize the camera pose estimates using VGGT [12]. Second, we refine all camera poses of the two external cameras from all timesteps jointly by aligning rendered robot geometry to observed depth using the recorded robot joint states.

Camera Pose Initialization. Our goal is to estimate, for each externally mounted camera C_i, a single rigid transform $T_{C_i \leftarrow B} \in \mathrm{SE}(3)$ that maps 3D points from the robot base frame B into the camera frame and is fixed throughout a DROID episode. We denote the robot base frame by B, the wrist camera frame at time t by W_t, and the external cameras by C_i. A multi-view pose estimator (VGGT [12]) is applied to time-aligned images from the two external cameras and the wrist camera; it treats the first external camera at an initial timestep as the reference frame, and returns rigid transforms $T_{E_0 \leftarrow C_i}$ and $T_{E_0 \leftarrow W_t}$ that map points from each camera C_i or W_t into this reference frame E_0. Independently, forward kinematics applied to the robot joint states provide the pose of the gripper in the base frame, $T_{G_t \leftarrow B}$. For each physical robot (given by the recorded robot serial), we assume that the wrist camera is rigidly mounted relative to the end effector and reused across all episodes. We empirically found that this assumption is largely valid for robots used in DROID as the averaged transforms exhibit sub-millimeter alignment error with each other. Under this assumption, we can obtain a known gripper-to-wrist transform $T_{W \leftarrow G}$ for each robot. Using this transform and the forward-kinematics model, we obtain a time-varying wrist pose in the base frame,

$$T_{W_t \leftarrow B} = T_{W \leftarrow G}\, T_{G_t \leftarrow B}.$$

Combining this with the estimator's transform between the wrist and the reference external camera yields per-frame estimates of the reference camera pose in the base frame,

$$T^{(t)}_{E_0 \leftarrow B} = T_{E_0 \leftarrow W_t}\, T_{W_t \leftarrow B},$$

which we average over all valid wrist frames to obtain a single $T_{E_0 \leftarrow B}$. For any other external camera C_i, the estimator provides a relative transform $T_{E_0 \leftarrow C_i}$. We convert this to a base-frame extrinsic

$$T_{C_i \leftarrow B} = T_{C_i \leftarrow E_0}\, T_{E_0 \leftarrow B}, \quad T_{C_i \leftarrow E_0} = T^{-1}_{E_0 \leftarrow C_i},$$

so that all external cameras are expressed in the same robot base frame before the refinement stage.

Camera Pose Refinement. Starting from the initialized base-frame extrinsics $T_{C_i \leftarrow B}$, we jointly refine the poses of all external cameras using robot-depth reprojection. Let external cameras be indexed by i, timesteps by t, and let k index valid robot pixels after filtering (in front of the camera, within image bounds, deduplicated in the image plane, and with observed depth in the trusted range). For each camera i we optimize a small 6-DoF update on top of the initialization, parameterized by translation and rotation and scaled such that optimization can be done within a good numerical range. Given an observed depth value $d^{\text{obs}}_{i,t,k}$ at a valid robot pixel and the corresponding predicted depth $d^{\text{pred}}_{i,t,k}$ obtained by projecting robot surface points from the base frame through the current extrinsics $T_{C_i \leftarrow B}$, we define the robot-depth reprojection loss

$$L_{\text{robot-depth}} = \frac{1}{K} \sum_{i,t,k} \left| d^{\text{obs}}_{i,t,k} - d^{\text{pred}}_{i,t,k} \right|,$$

where the sum runs over all valid robot pixels across cameras and frames and K is their total count. We optimize the 6-DoF updates for all external cameras jointly using a first-order optimizer (100 iterations, learning rate 10^{-3}), and restrict supervision to robot pixels whose observed depth lies in a trusted range of $[0.3, 2.0]$ m. To ensure reliable gradients, we further require that each camera–frame pair contributes at least $2{,}000$ valid robot points; frames that fail this criterion for any external camera are discarded before refinement. With these procedures, we can accurately label camera poses for around 60% of all episodes recorded in DROID. Quantitative metrics are reported in the main text.

A.5.3. Benchmark Metrics for 3D Annotation

Depth Reprojection. For each scene and calibration variant, we render the robot mesh into each external camera using the candidate extrinsics and recorded joint states, discard depth outside $[0.3, 2.0]$ m or out-of-bounds robot pixels, and compute an L1 depth reprojection loss over valid robot pixels. The per-frame losses are averaged over valid pixels and frames to produce a point-weighted value reported per scene and then aggregated into cumulative curves.

Two-view F1 @ 5/20 mm. For each external camera, we back-project valid depth into 3D using corresponding intrinsics/extrinsics, remove robot pixels, and crop to workspace. Given the resulting paired point clouds, we compute precision, recall, and F1 via symmetric nearest-neighbor matching: a point in cloud A (resp. B) is a true positive if its nearest neighbor in B (resp. A) lies within the threshold (5 mm or 20 mm); otherwise it contributes to the false-positive/false-negative count. Metrics are accumulated over frames, normalized by the number of valid points per view, and then aggregated per scene into the cumulative counts shown in Figure 6. Scenes with missing depth/extrinsics for a given combo are omitted from evaluation for that combo.

A.5.4. Occlusion-Aware Tracking in 3D

Given depth and camera poses, we herein describe how we can obtain 3D point flows. We describe the following in order: (i) filtering clips based on robot motion, (ii) tracking visible points within each retained clip, and (iii) postprocessing the resulting 3D trajectories.

Clip Filtering. We slice each episode into overlapping clips of length $F{=}16$ frames with stride $s{=}1$. Each clip covers approximately one second of the episode. Clips are retained if either the gripper changes state within the clip or end-effector motion exceeds thresholds. Thresholds depend on whether the gripper is predominantly open or closed within the clip: position/rotation thresholds are $(0.005\,\mathrm{m},\ 0.10\,\mathrm{rad})$ when open and $(0.002\,\mathrm{m},\ 0.05\,\mathrm{rad})$ when closed; either exceeding suffices to keep a clip, and any change in gripper state also keeps the clip.

Tracking. After obtaining depth and camera poses and slicing episodes into short clips, we perform dense tracking to obtain 3D scene flows on a per-clip basis. Clips cover roughly one-second windows and, depending on where they fall within a longer episode, may observe quite different regions of the workspace; tracking after clip selection ensures that each clip has its own consistent set of tracked regions and avoids mixing trajectories across widely separated time intervals. To improve efficiency and robustness, we perform tracking restricted to workspace and non-robot regions. To construct the workspace mask, we project a fixed 3D workspace volume to the image. To construct the non-robot mask, we render the robot's URDF and project the mesh to the image plane. We then track 2D points using 2D point trackers (CoTracker3 [13]) on the masked regions only, producing dense 2D trajectories. The tracker also outputs a per-point visibility mask over time; we store this visibility for each 2D trajectory so that occluded points can later be excluded from supervision after lifting to 3D. Because POINTWORLD is trained on 3D point flows rather than image-plane trajectories, we lift each tracked 2D point to a 3D world-space trajectory by back-projecting it with the corresponding depth, intrinsics, and extrinsics at each timestep. We next reconstruct 3D trajectories and apply clip-level postprocessing. For each frame, valid depth, intrinsics, and extrinsics back-project tracked pixels to world-frame 3D points with RGB. Tracks across time yield temporally consistent per-point trajectories. We store per-camera trajectories to avoid mixing viewpoints prematurely and keep per-point visibility and depth-valid flags that are later used to mask supervision.

Postprocessing. To further improve the quality of the obtained 3D point flows, we apply two postprocessing steps: DBSCAN-based outlier removal and per-frame normal estimation. For each clip we first remove spatial outliers using multi-scale DBSCAN clustering [14] across all external cameras: at each timestep, we run DBSCAN with radii $\varepsilon \in \{0.02, 0.05\}$ m and minimum core size 5, and mark point flows that are classified as outliers in more than 20% of frames as outliers to be discarded. From the remaining trajectories, we estimate normals per frame using local neighborhoods (up to $k{=}30$ neighbors within radius 0.1 m) and orient them toward the camera, followed by a temporal consistency step that flips back-facing normals so that normal directions remain coherent over time.

A.6. BEHAVIOR-1K Data Generation

BEHAVIOR-1K (B1K) [15] is a large-scale benchmark of everyday household activities in photorealistic simulation built on NVIDIA Isaac Sim. As part of the 2025 BEHAVIOR Challenge, it provides approximately $10{,}000$ human-teleoperated episodes (average length ≈ 6.6 minutes) across 50 tasks executed by a bimanual mobile robot (Galexea R1 Pro). We replay these episodes in the original simulator and attach three virtual cameras (left shoulder, right shoulder, and head) to extract short clips with meaningful interactions and dense 3D point flows, as detailed below.

A.6.1. Dataset Replay

For each BEHAVIOR-1K episode, we replay it in the simulator using the recorded sequence of environment states and actions. To prevent physics leakage and adhere to the original demonstrations, we iterate over the stored trajectory and, at every recorded step, load the corresponding simulator state and advance the simulator once with the recorded action. At every such step, we also render three external RGB-D cameras mounted on the robot: a left and right shoulder camera attached near the base, and a head-mounted camera. All three cameras have ground-truth intrinsics and extrinsics, and produce per-pixel depth, surface normals, and per-link segmentation in addition to RGB. All extrinsics are recorded in the robot base frame in the first timestep of each clip.

A.6.2. Clip Filtering

We aim to extract short clips of fixed length $F{=}11$ frames that contain meaningful interaction between the robot and the scene while discarding static or uninteresting intervals. To generate candidates, we slide an overlapping window of length F over each replayed episode at a fixed temporal stride; any window for which *all* external cameras have no visible, in-workspace scene objects—that is, no non-robot, non-ground meshes with valid depth inside the workspace bounds—is immediately discarded.

For each remaining candidate window, we maintain a set of *motion indicators* and *contact indicators* that are updated over the clip. Let M_g denote the event that at least one arm's end-effector exhibits sufficient translational or rotational motion over the clip or undergoes a change in gripper open/closed state, with thresholds that depend on whether the gripper is predominantly open or closed. Let M_j denote the event that any non-base robot joint moves more than a fixed threshold over the clip. Using the object trajectories described in Section A.6.3, we define M_o as the event that at least one object moves more than an object-movement threshold in position or orientation relative to its pose at the first frame of the clip.

From the ground-truth simulation state, we further construct contact indicators. Let C_t denote the event that any trunk or arm link experiences a nonzero contact impulse during the clip, and let C_f denote the event that any gripper finger link experiences contact. Clips that contain large simulator-induced discontinuities (such as scene resets) are filtered internally before applying the following logical criterion.

At the end of the horizon, a remaining candidate is accepted as a valid clip if and only if

$$\neg C_t \wedge \big((M_o \wedge M_j) \vee (M_o \wedge C_f) \vee (\neg M_o \wedge M_g \wedge M_j)\big). \quad (1)$$

The term $\neg C_t$ discards clips that contain trunk or arm collisions. The first disjunct in equation 1 retains clips where object motion is causally associated with non-base joint motion. The second disjunct retains clips where object motion primarily arises through gripper-finger contacts, which covers behaviors such as pushing an object with only base motion rather than arm motion (e.g., pushing a door by moving the base). The third disjunct retains "negative" clips in which the robot moves but no objects move, providing supervision on background dynamics and free-space motions.

A.6.3. 3D Point Flows from Simulation

For each accepted clip and each external camera, we construct a compact representation of 3D point flows that exploits three properties of the simulation setting: (i) the environment is composed of rigid objects decomposed into rigid links; (ii) we have access to ground-truth link-level instance segmentation in rendered images; and (iii) we can query the exact rigid trajectory of every link throughout the clip. At the first frame of the clip, we back-project depth for each visible link to obtain a set of surface points in that link's local frame, together with associated colors and normals, after filtering out background and robot meshes and enforcing workspace bounds in the robot base frame at the clip start. We then record the time-varying poses of all visible links and cameras in this same clip-start robot base frame. This factorized representation, local link points plus per-link trajectories, allows us to reconstruct exact per-point 3D trajectories for any clip while remaining far more storage-efficient than storing dense point clouds at every frame. Note that while we access ground-truth simulator state for obtaining ground-truth 3D point flows, the simulator state is never exposed to the model.

Operation	Description
Camera subsampling	Sample randomized calibrated RGB-D views per timestep and concatenate their 3D points into a single scene cloud.
Bounds filtering	Retain only scene points that stay within a workspace cube (approx. $[-3, 3]^3$ m) for the entire clip, dropping particles that ever exit the bounds.
Centering	Mean-center first-frame scene and robot points.
Image resize	Downscale RGB-D images to 320×180.
Voxel downsampling	Voxel-grid sampling at 1.5 cm; select one point per occupied voxel at $t{=}0$, and apply the same indices to all timesteps.
Multi-sphere cropping	Iteratively remove spheres of points far from the robot (up to three spheres, radii in $[0.10, 0.80]$ m with buffer 0.25 m) until the scene falls below the target budget.
Max scene / robot points	Randomly subsample scene points if more than $12\,000$ remain after cropping; robot points are capped at 500 by construction.
Random yaw	Uniform rotation about the vertical axis over $[-\pi, \pi]$.
Uniform scaling	Isotropic scaling with factor sampled uniformly from $[0.9, 1.1]$.
Random reflection	With probability 0.5, reflect the scene and robot across either the x- or y-axis.
Chromatic auto-contrast	Apply chromatic auto-contrast to RGB channels with probability 0.2 and blend factor up to 0.2.
Chromatic translation	Add a global RGB offset with magnitude 2% with probability 0.95.
Chromatic jitter	Add per-point RGB noise with standard deviation 2% with probability 0.95.

Table 1. **Data Preprocessing and Augmentations.**

Point set	Feature	Definition
Robot	Position $p_{t,j}^{\text{robot}}$	3D coordinates of robot points over time.
Robot	Color c_j^{robot}	Constant magenta color $(1, 0, 1)$ indicating robot identity, shared across timesteps.
Robot	Normal $n_{t,j}^{\text{robot}}$	Surface normals of robot points from the known robot URDF.
Robot	Gripper openness $\tilde{g}_t$	Scalar gripper open value per timestep, broadcast to all robot points.
Robot	Velocity $v_{t,j}^{\text{robot}}$	Per-point velocity from mid-point finite differences over $p_{t,j}^{\text{robot}}$ across time.
Robot	Acceleration $a_{t,j}^{\text{robot}}$	Per-point acceleration from mid-point finite differences over $v_{t,j}^{\text{robot}}$ across time.
Scene	Position $x_{0,i}$	3D coordinates of scene points at the first frame after preprocessing.
Scene	Color $c_{0,i}^{\text{scene}}$	RGB color of scene points at the first frame.
Scene	Normal $n_{0,i}^{\text{scene}}$	(Estimated) surface normals of scene points at the first frame.
Scene	Gripper openness sequence $g_{0:T-1}$	Sequence of gripper openness values over the context and prediction horizon, broadcast to every scene point.
Scene	Distance-to-robot $d_{0:T-1,i}$	For each timestep, distance from the first-frame position of scene point i to the closest robot point, stacked across time.

Table 2. **Per-Point Input Features.**

A.7. Model Training Details

Data Preprocessing and Augmentations. Here we describe the data preprocessing and augmentations used in our experiments. Each training sample fuses calibrated RGB-D views before passing through workspace filtering, centering, and deterministic voxel sampling with multi-sphere cropping so that the fused cloud respects fixed budgets for scene and robot points. Geometric augmentations consist of random yaw rotations, isotropic scaling, and reflections; photometric augmentations apply auto-contrast, global color shifts, and per-point jitter to the RGB channels. For evaluation, we ensure the pipeline to be fully deterministic and disable all augmentations.

Per-Point Input Features. Here we describe the per-point features produced as part of the data pipeline, prior to their consumption by the model. Details are listed in Table 2. Robot features stack positions, surface normals, a gripper scalar, and velocity and acceleration terms:

$$\phi_{t,j}^{\text{robot}} = \left[p_{t,j}^{\text{robot}},\, c_j^{\text{robot}},\, n_{t,j}^{\text{robot}},\, \tilde{g}_t,\, v_{t,j}^{\text{robot}},\, a_{t,j}^{\text{robot}}\right],$$

where $p_{t,j}^{\text{robot}}$ and $n_{t,j}^{\text{robot}}$ are position and normal, c_j^{robot} is a fixed color tag, $\tilde{g}_t$ is the normalized gripper openness, and $(v_{t,j}^{\text{robot}}, a_{t,j}^{\text{robot}})$ come from mid-point finite differences across the horizon with zero-velocity boundary conditions at the first and last timestep, i.e., we assume the robot is stationary at the boundaries of each model window. Scene features are computed for only the first frame $t{=}0$ and combine positions, colors, estimated normals, gripper openness sequence, and distances to the nearest robot point:

$$\phi_i^{\text{scene}} = \left[x_{0,i},\, c_{0,i}^{\text{scene}},\, n_{0,i}^{\text{scene}},\, g_{0:T-1},\, d_{0:T-1,i}\right],$$

where $c_{0,i}^{\text{scene}}$ and $n_{0,i}^{\text{scene}}$ are the RGB color and normal at the first frame, $g_{0:T-1} \in \mathbb{R}^T$ is the sequence of gripper openness values over the context-plus-prediction horizon broadcast to every scene point, and $d_{t,i}$ is the distance from scene point i to the closest robot point at timestep t,

$$d_{t,i} = \min_j \left\| x_{0,i} - r_{t,j} \right\|_2, \qquad d_{0:T-1,i} \in \mathbb{R}^T.$$

The distance field $d_{0:T-1,i}$ is obtained from nearest-neighbor queries between first-frame scene points and robot points at every timestep.

3D Scene Featurization with DINOv3. Prior to the point cloud backbone, POINTWORLD uses a 2D scene encoder based on DINOv3 ViT-L/16 by aggregating its multi-layer features. To featurize the 3D scene points with the image-based encoder, the first-frame scene coordinates $x_{0,i} \in \mathbb{R}^3$ are projected into each chosen camera. For camera c with intrinsics K_c and extrinsics (R_c, t_c) we form $\tilde{u}_{c,i} = K_c(R_c x_{0,i} + t_c)$ and obtain the pixel coordinate as

$$u_{c,i} = \left[\tilde{u}_{c,i}^{(1)}/\tilde{u}_{c,i}^{(3)},\, \tilde{u}_{c,i}^{(2)}/\tilde{u}_{c,i}^{(3)}\right]^\top.$$

The same intrinsics and extrinsics support a depth-consistency mask that compares the projected depth of each point with the given depth image, so only views whose discrepancy is below a few millimeters contribute features.

For each visible point-camera pair (i, c), DINOv3 patch tokens are sampled at $u_{c,i}$ by bilinear interpolation on the patch-token grid, using a coordinate mapping that aligns token centers with pixel centers. Let $f_{c,i} \in \mathbb{R}^{D_{\text{patch}}}$ denote the concatenated multi-layer patch feature for point i in camera c, and let $m_{c,i} \in \{0, 1\}$ indicate visibility and depth consistency. Features are aggregated across cameras by averaging over the contributing views,

$$f_i = \frac{1}{\max\left(1, \sum_c m_{c,i}\right)} \sum_c m_{c,i}\, f_{c,i}.$$

The averaged token is mapped to the backbone width (256 channels) by a learned projection and fused with a separately projected version of the raw scene features from Table 2; layer normalization is applied to each stream before concatenation, and a final linear layer produces the per-point embedding supplied to the dynamics backbone. The 2D encoder is kept frozen during training and evaluation.

Visibility-Aware Supervision. For real-world domains, we restrict training on 3D point trajectories to correspondences that are both geometrically and photometrically reliable. The annotation pipeline supplies per-point visibility (from 2D trackers [13] on real data and from ground-truth simulator state on synthetic data) together with per-pixel depth-validity; both signals are propagated to the lifted 3D trajectories and stored as binary flags per scene point and timestep. During training, we construct a per-timestep mask that selects scene points that are visible in the camera view and have valid depth support. The weighted dynamics objective from Section 3 is then evaluated only over this subset of correspondences (points filtered out receive zero loss weight), so that gradients are driven by non-occluded, depth-valid 3D flows. For simulation domains, where trajectories and depth are noise-free and occlusions are explicitly modeled, all scene points contribute to the loss.

Training Configuration. We train the 1B-parameter version of POINTWORLD on both BEHAVIOR-1K and DROID, with configuration and PointTransformerV3 (PTv3) design summarized in Table 3 and Table 4, respectively. For the main experiments in the paper, training configuration and PTv3 design are summarized in Table 5 and Table 6.

Aleatoric Uncertainty on Simulation Data. When training on mixtures of real and simulated domains, directly learning per-point uncertainty everywhere can collapse the model because simulated trajectories are noise-free. In the objective from Section 3 the residual term for point i at step k is weighted as $w_{k,i}\,\rho_\delta(\hat{\mathbf{P}}_{t+k,i} - \mathbf{P}_{t+k,i})\,\mathrm{e}^{-s_{k,i}} + w_{k,i}\,s_{k,i}$. For vanishing residuals (typical in simulation), minimizing the loss drives $s_{k,i}$ toward $\log \rho_\delta(\cdot)$. Since $\rho_\delta(\cdot)$ approaches zero, the optimal $s_{k,i}$ becomes a large negative number, i.e., the predicted variance $\sigma^2_{k,i} = \exp(s_{k,i})$ collapses toward zero. As a consequence, $\mathrm{e}^{-s_{k,i}}$ explodes, so any small numerical discrepancy in simulated residuals produces excessively large gradients that overwhelm the real-data contributions and destabilize joint training. To stabilize training, we treat aleatoric variance on simulation domains as a constant: the uncertainty head is trained normally on real data, but for simulated domains its log-variance is replaced by a batch-wise constant that matches the average variance observed on real samples (or a small fixed value when only simulation is present). This preserves heteroscedastic weighting where it is most useful (real, noisy supervision) while preventing the model from exploiting the uncertainty head to down-weight clean simulated gradients.

Setting	Value
Optimizer	AdamW
Learning rate	1×10^{-4}
Epochs	300
Weight decay	10^{-2}
Global batch size	1920 sequences
Gradient clipping	Global ℓ_2 norm capped at 5
Loss	Huber loss with $\delta = 5.0$ with movement weighting and aleatoric uncertainty
Prediction horizon	10 steps
Training GPUs	128 NVIDIA H100 GPUs
Training time	20 days

Table 3. **Training Configuration for POINTWORLD-1B.**

Component	Values
Grid size	1.5 cm
Encoder depth	(4, 4, 8, 8, 12, 12, 4)
Encoder channels	(256, 384, 384, 512, 512, 768, 1024)
Encoder heads	(8, 12, 12, 16, 16, 24, 32)
Encoder stride	(1, 2, 2, 2, 2, 2, 2)
Encoder patch size	(256, 256, 256, 256, 256, 256, 256)
Decoder depth	(4, 4, 4, 4, 4, 4)
Decoder channels	(256, 384, 384, 512, 512, 768)
Decoder heads	(8, 12, 12, 16, 16, 24)
Decoder patch size	(256, 256, 256, 256, 256, 256)

Table 4. **PointTransformerV3 (PTv3) Architecture for POINTWORLD-1B. Encoder and decoder configurations are ordered by stage.**

Setting	Value
Optimizer	AdamW
Learning rate	1×10^{-4}
Epochs	200
Weight decay	10^{-2}
Global batch size	176 sequences
Gradient clipping	Global ℓ_2 norm capped at 5
Loss	Huber loss with $\delta = 5.0$ with movement weighting and aleatoric uncertainty
Prediction horizon	10 steps
Training GPUs	8 NVIDIA H100 GPUs
Training time	7 days

Table 5. **Training Configuration for POINTWORLD-411M.**

Component	Values
Grid size	1.5 cm
Encoder depth	(4, 4, 4, 8, 8, 12, 4)
Encoder channels	(256, 256, 256, 384, 384, 512, 768)
Encoder heads	(4, 4, 4, 8, 8, 16, 24)
Encoder stride	(1, 2, 2, 2, 2, 2, 2)
Encoder patch size	(256, 256, 256, 256, 256, 256, 256)
Decoder depth	(2, 2, 2, 2, 2, 2)
Decoder channels	(256, 256, 256, 384, 384, 512)
Decoder heads	(4, 4, 4, 8, 8, 16)
Decoder patch size	(256, 256, 256, 256, 256, 256)

Table 6. **PointTransformerV3 (PTv3) Architecture for POINTWORLD-411M. Encoder and decoder configurations are ordered by stage.**

A.8. DROID Evaluation Protocol

Following the protocol in Section 5, we measure per-sequence losses and aggregate them into dataset-level summaries. Alongside the overall per-point, per-timestep ℓ_2 distance, we report the same metric separately on moved and static points, since movers form a minority of the points but dominate perceived quality. We use these metrics directly for simulation data (BEHAVIOR-1K) since they are noiseless. However, for real-world data (DROID), we further apply expert confidence filtering to obtain filtered metrics. Details are described below.

Expert Confidence Filtering. Although the mover-ℓ_2 score highlights the behavior we care about most, imperfect real-world annotations mean that a noticeable fraction of mover points correspond to outliers or background clutter, because those points tend to have large movement magnitudes due to unstable depth estimation. During training, the aleatoric uncertainty regularization down-weights those points, but at evaluation time different models produce their own uncertainty predictions, making comparisons challenging. To obtain a model-agnostic notion of trustworthy ground-truth, we train an expert model only on the evaluation split with uncertainty predictions, convert the predicted variance into a per-point confidence in $[0, 1]$, and threshold this per-timestep per-point confidence at the 0.8 quantile over all points. Points below this confidence are treated as low-confidence outliers. We voxelize these low-confidence sets in world coordinates using the training grid size g and cache the resulting voxel grids for each evaluation sample so that the same filtering masks can be reused across subsequent evaluation runs and model variants. Note that the expert model is only used to compute the low-confidence voxel grids and does not share any training data or parameters with any evaluated models.

Filtered Evaluation. To evaluate a model, for each sample, we first reconstruct world-coordinate voxel indices of scene points and then determine whether each point lies inside a precomputed low-confidence voxel. This yields a binary filter mask so that only high-confidence points at high-confidence timesteps (deemed by the shared expert model) contribute to the filtered metrics.

Mover/Static Splits. Let $\hat{P}_{t,i}$ and $P_{t,i}$ denote predicted and ground-truth 3D positions. We compute per-point error $e_{t,i} = \|\hat{P}_{t,i} - P_{t,i}\|_2$ and report

$$\ell_2 = \frac{1}{T} \sum_t \frac{1}{|V_t|} \sum_{i \in V_t} e_{t,i},$$

where V_t denotes valid points at timestep t. Mover-ℓ_2 and static-ℓ_2 use the same definition but restrict V_t to moved or static points identified from the ground-truth trajectories via a small displacement threshold.

A.9. Real-Robot Experiment Details

Real-robot experiments use a 7-DoF Franka arm equipped with a 3D-printed fin ray gripper [16, 17]. The robot is mounted on a wheeled, non-motorized base for in-the-wild deployments. Since POINTWORLD is trained on data containing Robotiq 2F-85 and Galexea R1 Pro grippers, the fin ray gripper geometry remains fully unseen by the model, illustrating cross-gripper geometry generalization. Since the pipeline predicts 6-DoF end-effector poses, we run position control at 20 Hz: each predicted target pose is clipped to a predefined workspace, then linearly interpolated from the current pose with steps of 5 mm in translation and 1° in rotation. For every interpolated pose, inverse kinematics (PyBullet solver) produces target joint positions that are tracked with the Deoxys joint-impedance controller [18]. We use one RealSense D435 mounted on the left shoulder of the robot to capture RGB and the stereo IR images. The stereo IR images are used to estimate the metric depth using FoundationStereo [11], given known baseline and camera intrinsics.

A.9.1. Model-Based Planning

In this work, we use a single pre-trained POINTWORLD as the dynamics model. The model is pre-trained jointly on both real-world and simulated data. We use a sampling-based model-predictive path integral (MPPI) controller that samples action sequences around a zero-initialized nominal using cubic splines with $n_{\text{knots}}=4$ and degree 3. Noise scales are scheduled between $\sigma_{\min}=0.05$ and $\sigma_{\max}=0.50$ (in normalized action units). Each refinement iteration draws 256 samples; importance weights use temperature $\beta=0.05$, and the nominal is updated with an exponential moving average (EMA $= 0.9$). We perform planning for 30 steps into the future, and the horizon is chunked to match the prediction window of the dynamics model. We perform 20 refinement iterations. The planning time is typically around a few seconds depending on task complexity and specific model size variant used. While we do not perform replanning in this work, replanning can be done at a real-time frequency by warm-starting from the previous nominal trajectory.

A.9.2. Task Specification

We specify tasks through a GUI tool that allows users to select object masks using SAM2 [19] and specify target positions in the world frame. We find this simple objective as a unified interface for specifying diverse real-world tasks including rigid pushing, deformable manipulation, articulated-object interaction, and tool use. Following common practices in reward design [20], we add a mild end-effector proximity term to encourage exploration in the object's neighborhood without prescribing a particular contact pattern. For deformable and tool-use tasks we begin from a pre-grasped configuration so that subsequent motion primarily probes deformable dynamics and object-object contacts. All tasks share same control regularization comprising SE(3) path-length penalties and IK-based reachability residual.

A.9.3. Evaluation Protocol

We conduct evaluations on the following tasks: rigid pushing (tissue box, book), deformable manipulation (scarf fold, pillow place), articulated manipulation (microwave open, drawer close), and tool use (duster sweep, broom sweep). Each task is evaluated with ten randomly sampled initial configurations. The configurations are sampled prior to evaluation and verified to be kinematically feasible for the robot. For each trial, a human operator restores the scene to the designated configuration and triggers execution. We consider the trial successful if the task objective is met. Otherwise we declare failure. If the optimization produces a solution that is considered unsafe for execution, the trial is considered failure too. The success rates are reported in the main paper.

A.9.4. Effect of Training Mixture

Beyond the quantitative success rates for real-world deployment, we observe interesting qualitative traits when using different variants of POINTWORLD pre-trained on different data mixtures. We empirically observe that models trained only on real data tend to be conservative: a common failure mode is for scene points to remain static even when the robot establishes contact, which we attribute to heavy regularization coping with annotation noise. On the other hand, models trained only on simulation data excel on rigid objects but frequently mis-segment cluttered real scenes implicitly, causing background points to move together with the target. Models trained on both real and simulated domains yield the most balanced behavior in practice, combining realistic contact handling with the ability to generalize to novel real-world scenes. A systematic study of how training-mixture design shapes deployment-time behavior, e.g., by varying real/sim proportions or task/domain coverage under controlled conditions, remains an important direction for future work.

A.10. Additional Figures

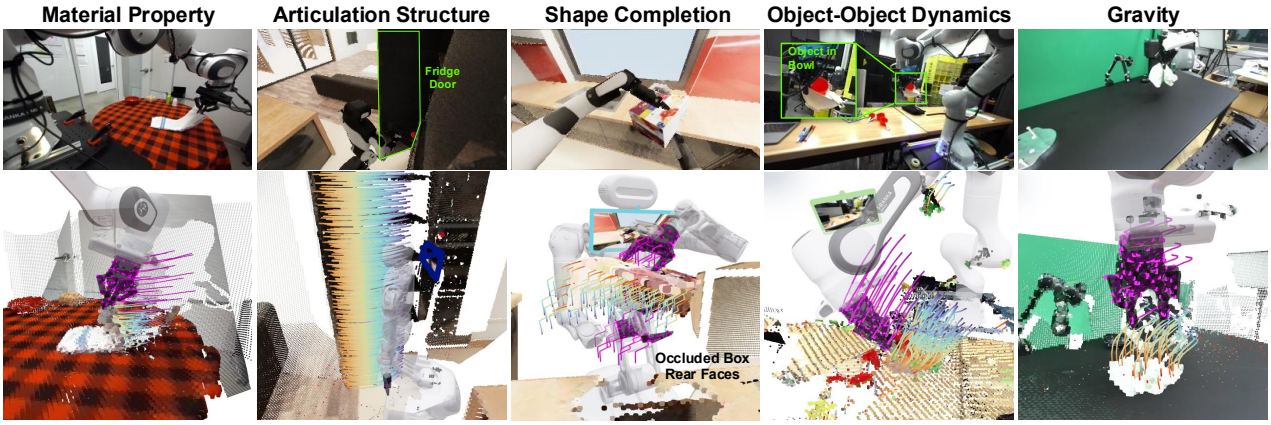

Figure 4. **Rich Supervision of 3D World Modeling for Physical Interactions**, when conditioned on 3D robot point flows and partial observable RGB-D. The 3D world modeling objective enjoys dense pixel-level supervision while encoding a wide range of capabilities central to robotic manipulation. To predict full-scene evolution, the model must implicitly segment objects, infer material or articulation, perform shape completion for contact reasoning, propagate robot-object interactions for object-object dynamics, and account for gravity in a single forward pass of the learned model.

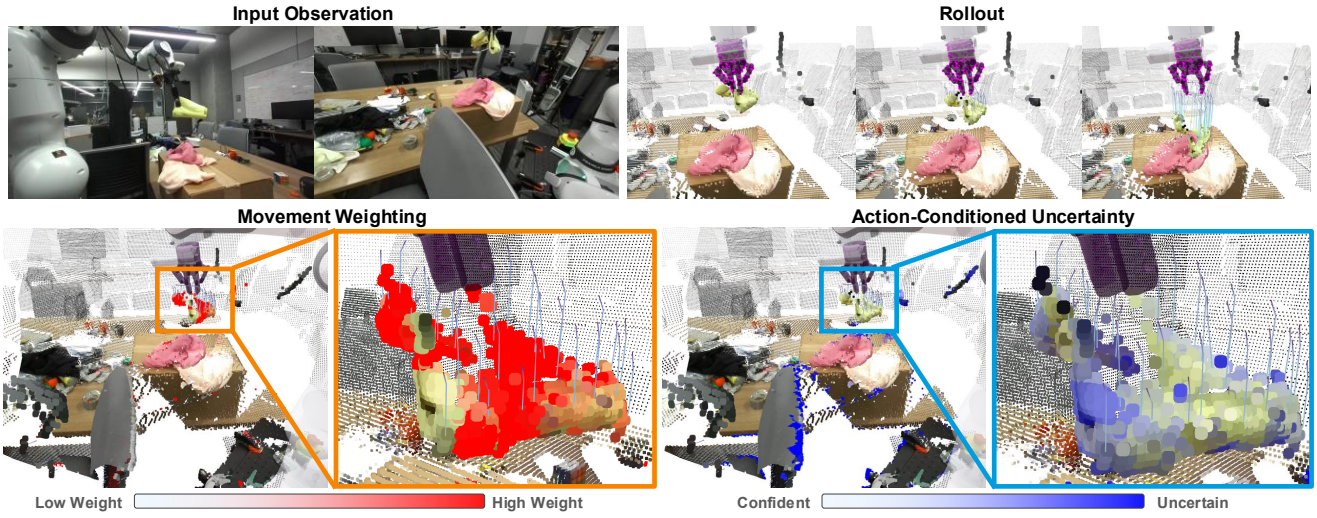

Figure 5. **Movement Weighting and Uncertainty Regularization**, where the robot releases and drops a yellow cloth. (**Bottom Left**) The movement weighting, used in the training objective, effectively biases the training towards scene points that are moving at each timestep, computed with the ground-truth flows. (**Bottom Right**) The uncertainty value, predicted by the model without any ground-truth, regularizes training to prevent overfitting to points that have unreliable ground-truth. Intriguingly, we observe that it also emerges to capture action-conditioned uncertainty arising from the object's physical properties (e.g., larger variability along the edge of the cloth).

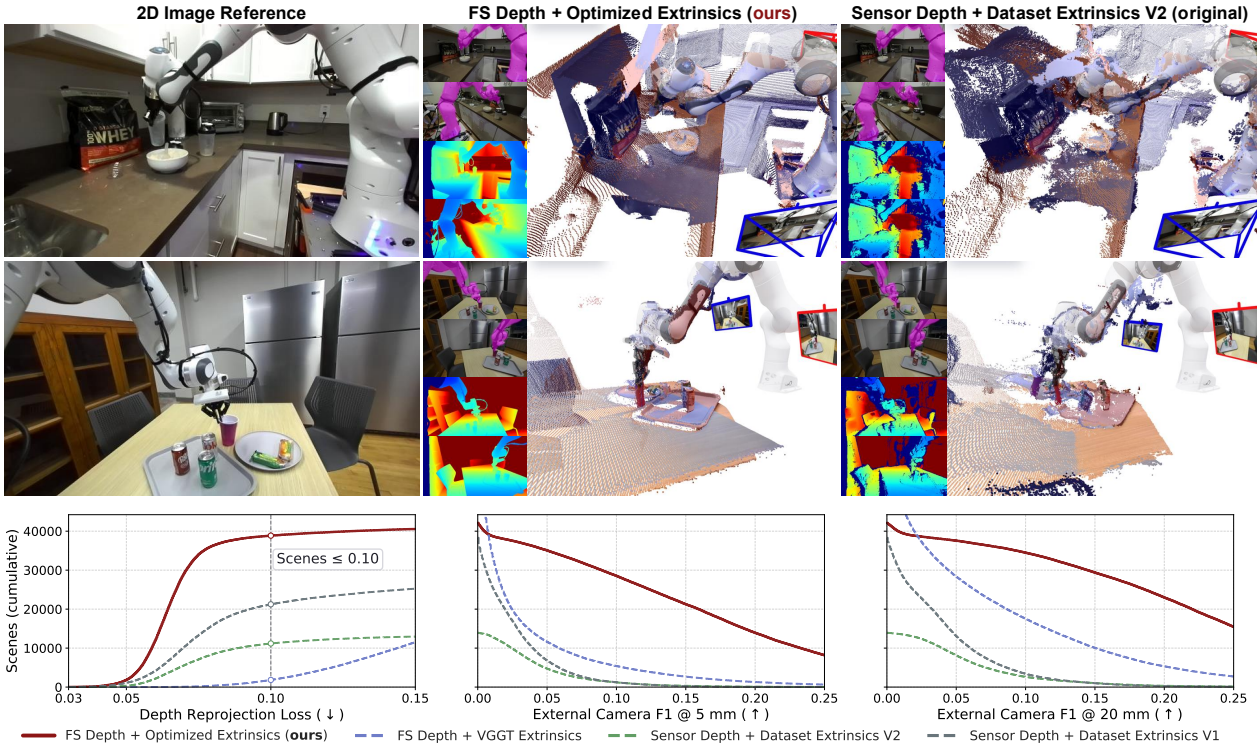

Figure 6. **3D Annotation Quality and Comparisons.** FS denotes FoundationStereo [11]; Dataset Extrinsics V1 and V2 are the two DROID extrinsics releases. **(Top)** Compared to DROID releases, our pipeline yields substantially higher-quality depth and camera pose calibration, resulting in more accurate robot mask overlays and better aligned point clouds. **(Bottom)** We further compute depth reprojection loss (differences between analytical and observed depth of robot surface), and F1 scores of point cloud alignment. We observe purely leveraging existing models (FS, VGGT) are insufficient, and V2 extrinsics improve over V1 by filtering out scenes with poor point cloud alignment but result in significantly lower scene counts. In contrast, our annotation pipeline retains substantially more scenes below 0.10 depth-loss criterion and dominates all metrics.

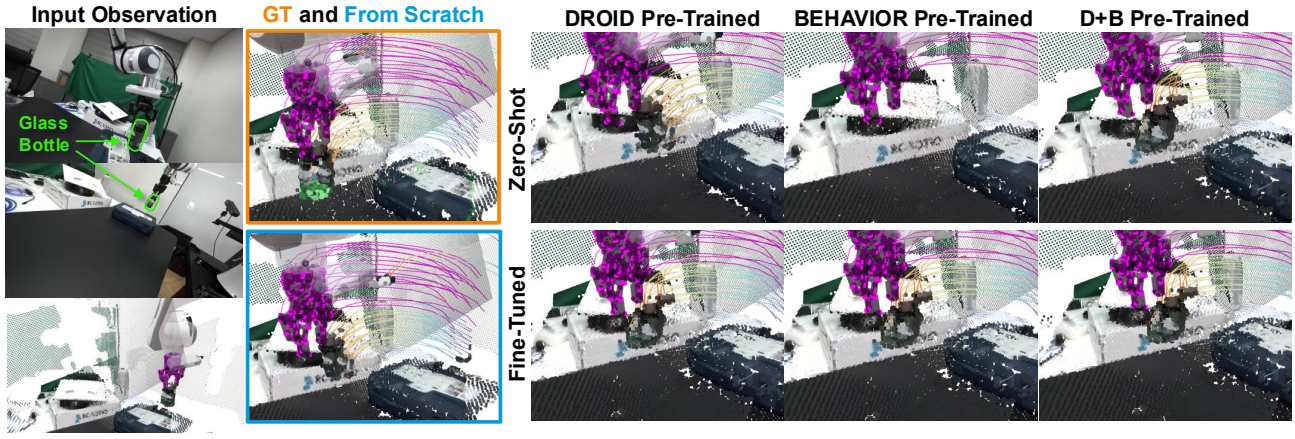

Figure 7. **Zero-Shot and Finetuned Generalization to Held-Out Real-World Scenes**, where the robot transports a reflective glass bottle. POINTWORLD pre-trained on DROID or jointly on DROID and BEHAVIOR (D+B) can zero-shot generalize to unseen environments and motion from a held-out DROID lab scene, closing the gap to the specialist trained on that lab's data. POINTWORLD pre-trained only on simulation data fail to generalize zero-shot. Further finetuning yields more accurate trajectories of grasped objects.

A.11. Additional 3D Annotation Examples

Interactive visualizations available at project website.

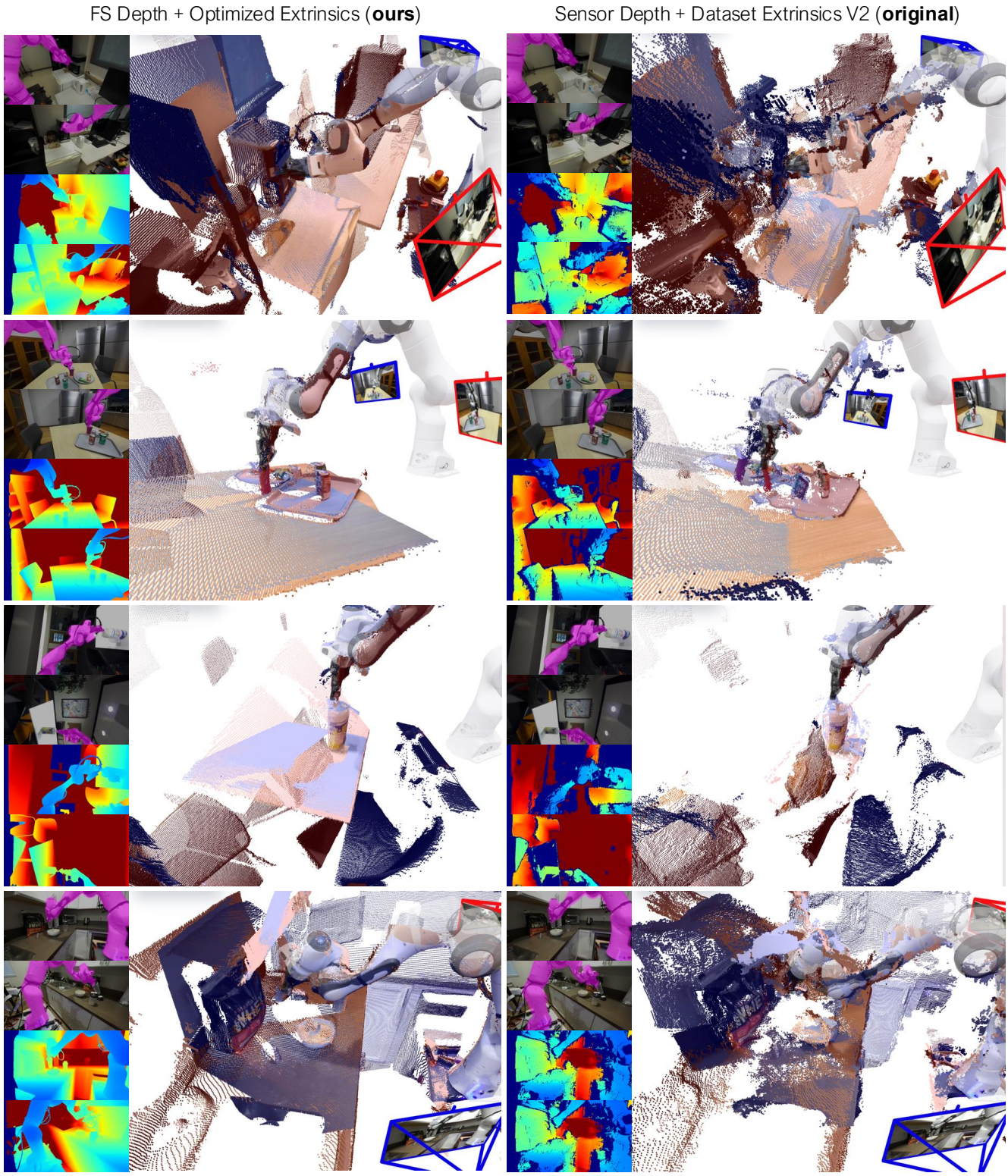

Figure 8. **DROID 3D annotations**, including robot-overlaid RGBs, depths, point clouds, and comparisons to original dataset.

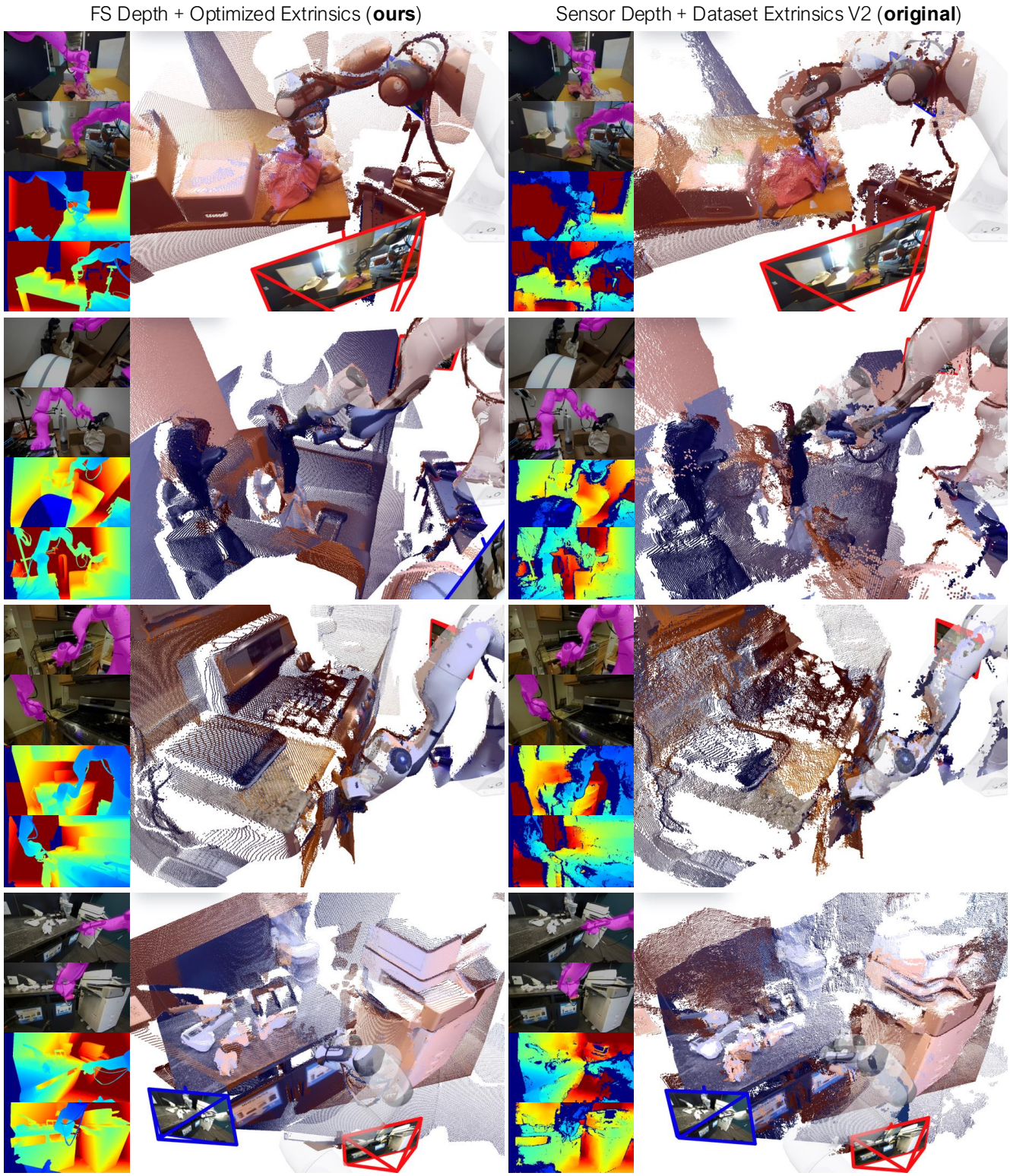

Figure 9. **DROID 3D annotations**, including robot-overlaid RGBs, depths, point clouds, and comparisons to original dataset.

A.12. Additional Model Rollouts

Interactive visualizations available at project website.

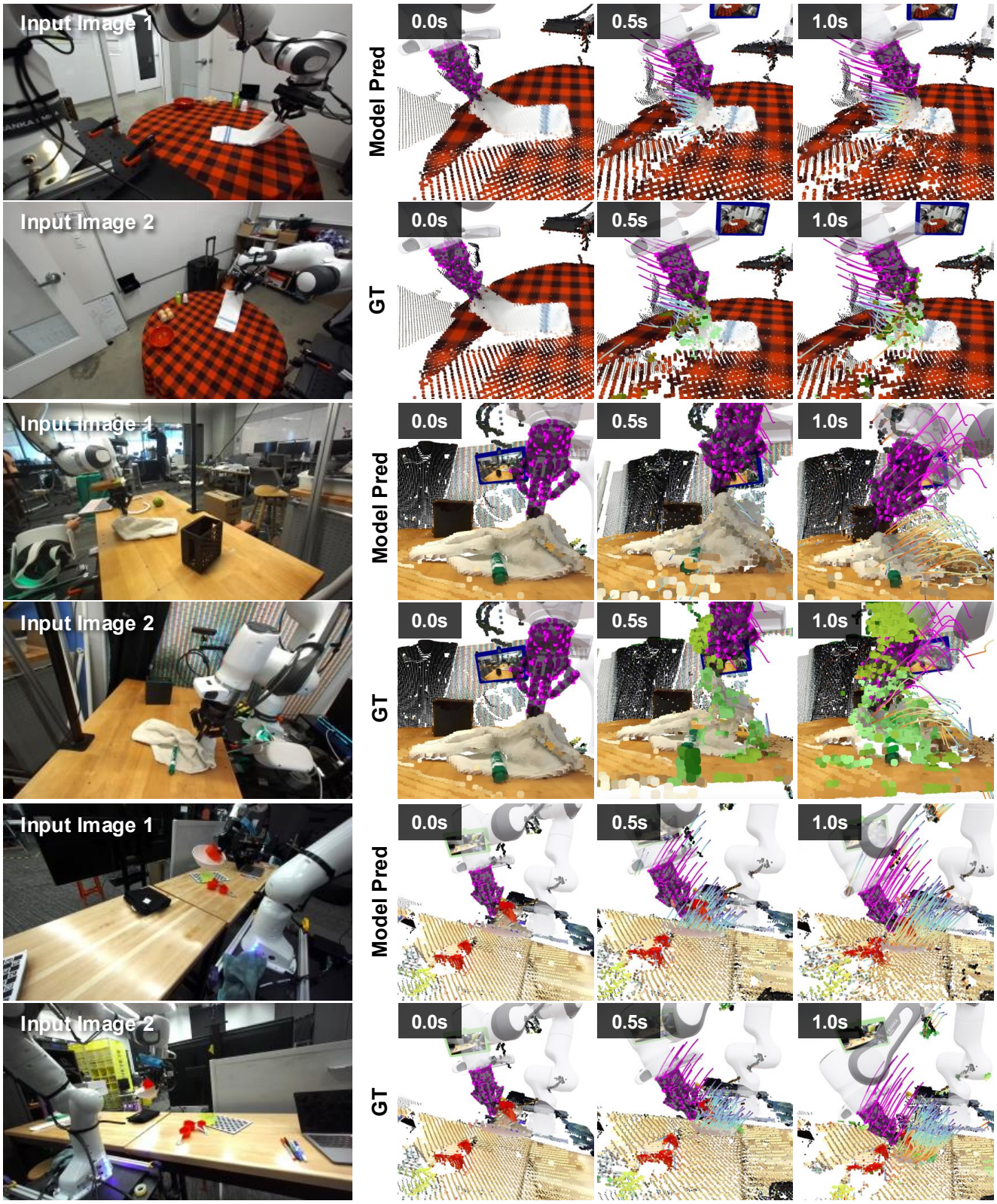

Figure 10. **DROID Unseen Rollouts**, including deformable manipulation, robot-object interactions, and object-object interactions.

Figure 11. **DROID Unseen Rollouts**, including deformable manipulation, and grasping behaviors.

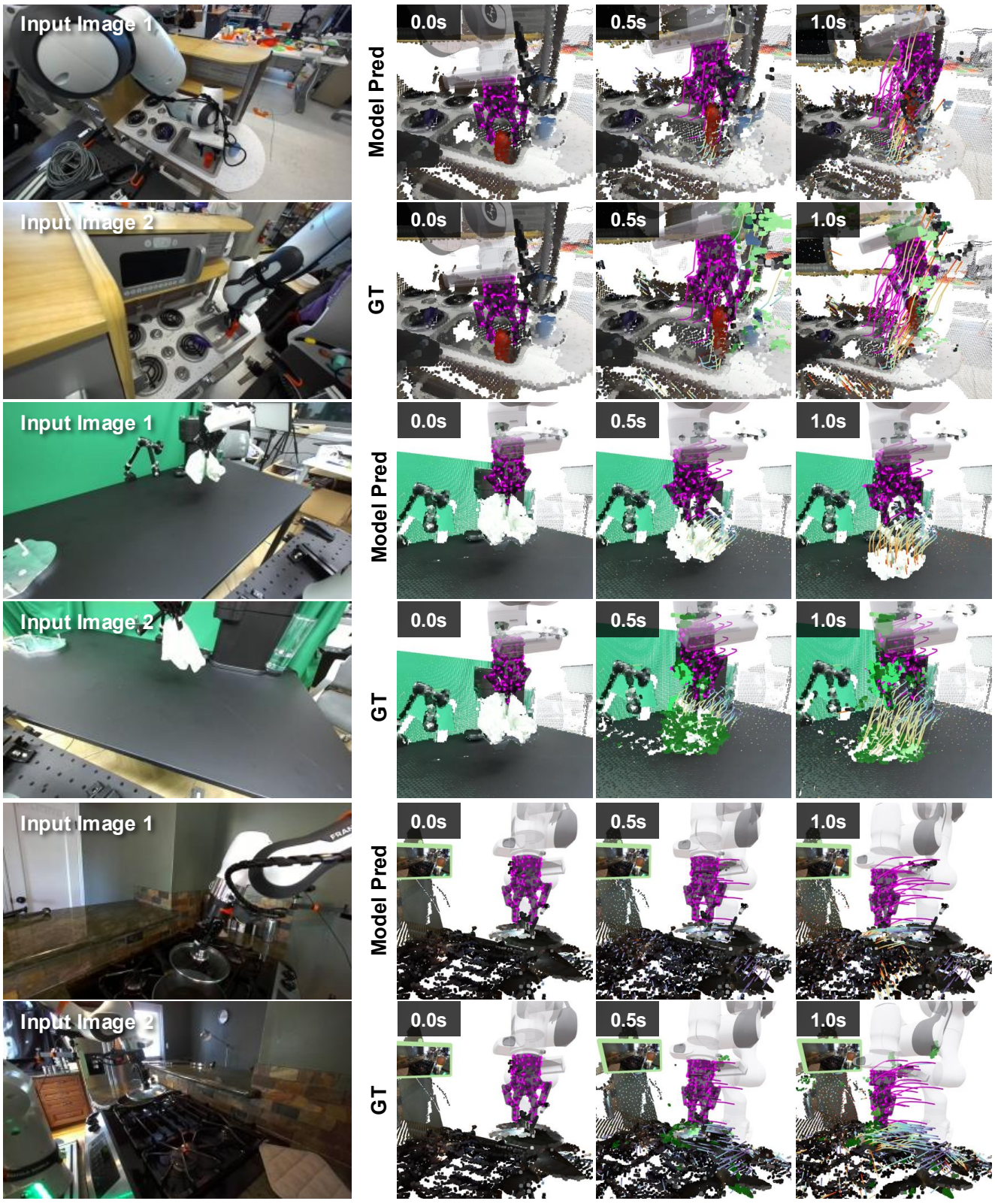

Figure 12. **DROID Unseen Rollouts**, including grasping behaviors, gravity effects, and glass objects.

Figure 13. **BEHAVIOR-1K Unseen Rollouts**, including constrained bimanual lifting, gravity effects (dropped laptop), object-object interactions (laptop v.s. table), and articulated manipulation (fridge).

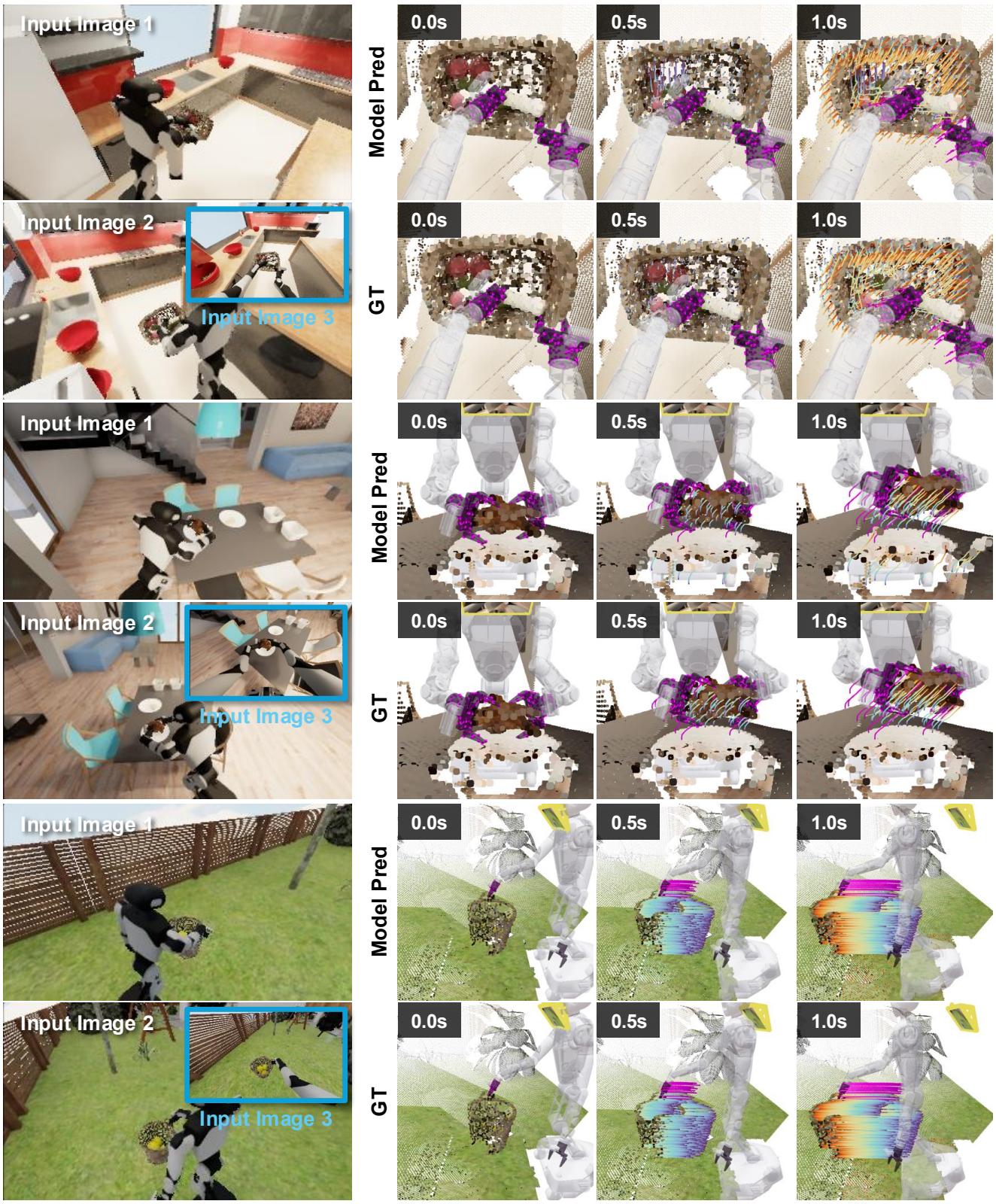

Figure 14. **BEHAVIOR-1K Unseen Rollouts**, including object-object interactions (within basket), gravity effects (within basket), and whole-body behaviors.

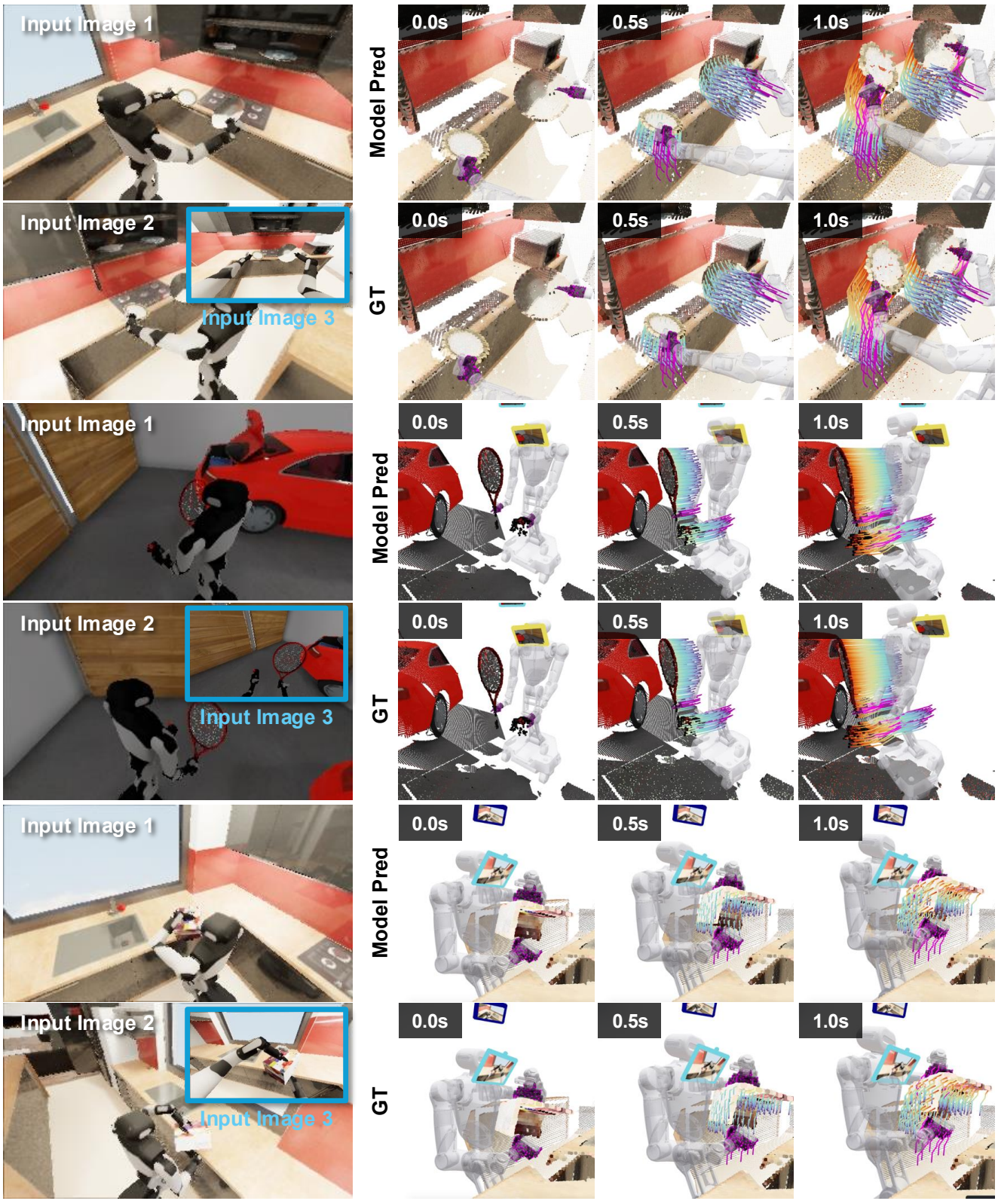

Figure 15. **BEHAVIOR-1K Unseen Rollouts**, including bimanual manipulation, whole-body behaviors, and implicit shape completion.

References

[1] Brent Yi, Chung Min Kim, Justin Kerr, Gina Wu, Rebecca Feng, Anthony Zhang, Jonas Kulhanek, Hongsuk Choi, Yi Ma, Matthew Tancik, and Angjoo Kanazawa. Viser: Imperative, web-based 3d visualization in python, 2025. 2

[2] Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017. 4

[3] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024. 4

[4] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004. 4

[5] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. 4

[6] Bernhard Kerbl, Georgios Kopanas, Thomas LeGendre, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 4

[7] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020. 4

[8] Sizhe Lester Li, Annan Zhang, Boyuan Chen, Hanna Matusik, Chao Liu, Daniela Rus, and Vincent Sitzmann. Controlling diverse robots by inferring jacobian fields with deep networks. *Nature*, pages 1–7, 2025. 4

[9] Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos. *arXiv preprint arXiv:2503.17973*, 2025. 4

[10] Alex Khazatsky, Karl Pertsch, Ashvin Nair, Ajay Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Madhav Srirama, Annie Chen, Benjamin Ellis, Patrick Fagan, Joseph Hejna, Maria Itkina, Lucille Lepert, Henry Ma, Alex Miller, Guanzhi Wu, Suneel Belkhale, Anish Dass, Jeongseok Ha, Ayush Jain, Angelica Lee, Youngwoon Lee, Nicole Memmel, Jeongho Park, Ilya Radosavovic, Rose Wang, Xuchen Zhan, Michael Black, Chi Chi, Landon Hatch, Jinqiang Lin, Zhenjia Lu, Jean Mercat, Ali Rehman, Pratyusha Sanketi, Akshara Sharma, Stacey Simpson, Quan Vuong, Pranav Walke, Blake Wulfe, Chih-Yuan Xiao, Brian Yang, Arman Yavary, Tony Zhao, Cem Agia, Parv Baijal, Alec Castro, Huan Chen, Tao Chen, Jen Jen Chung, Joshua Drake, Matthew Foster, Zhe Gao, Andres Herrera, Jeongwon Heo, Andy Hsu, Siyan Hu, Gabriel Jackson, Brian Le, Canyu Li, Hugo Lin, Donghun Ma, Avinash Maddukuri, Mihir Mirchandani, John Morton, Duc Nguyen, Brian O'Neill, Vincent Scalise, Jacob Seale, Doyeon Son, Yeming Tian, Quang Tran, Henry Wang, Andy Wu, Ho Chit Billy Xie, Juncheng Yang, Xiaolong Yin, Wenlong Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Anchit Gupta, Dinesh Jayaraman, Joseph J. Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Kuan-Ting Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems XX*, 2024. 5

[11] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5249–5260, 2025. 5, 13, 15

[12] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. 5

[13] Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6013–6022, 2025. 6, 9

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996. 6

[15] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024. 7

[16] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023. 13

[17] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024. 13

[18] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. *arXiv preprint arXiv:2210.11339*, 2022. 13

[19] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichten-

hofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 13

[20] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A Kahrs, et al. Mujoco playground. *arXiv preprint arXiv:2502.08844*, 2025. 13